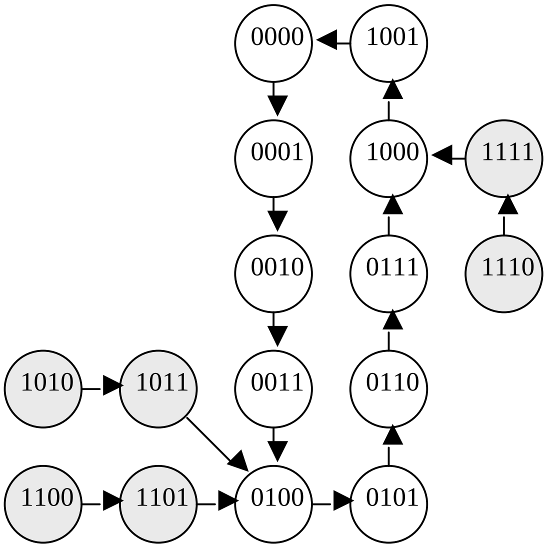
University of California at Berkeley
College of Engineering
Department of Electrical Engineering and Computer Sciences

EECS150 J. Wawrzynek
Spring 2000 E. Caspi

Homework #11 – Solution

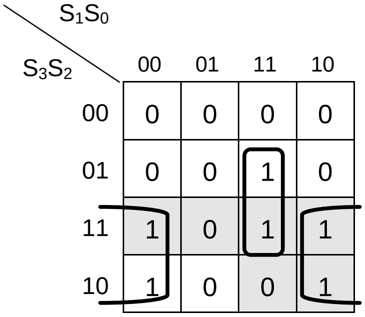
7.10 The BCD counter cycles in states 0-9. The self-starting transitions from states 10-15 (shown in gray) are chosen to make the K-map logic optimization convenient.



State Transition Diagram

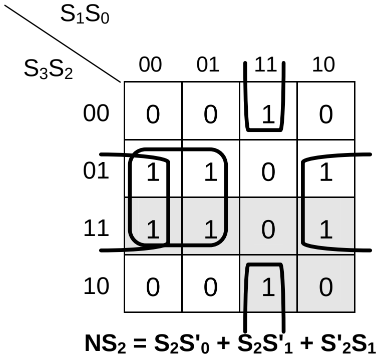
State				Next State			
S_3	S_2	S_1	S_0	NS_3	NS_2	NS_1	NS_0
0	0	0	0	0	0	0	1
0	0	0	1	0	0	1	0
0	0	1	0	0	0	1	1
0	0	1	1	0	1	0	0
0	1	0	0	0	1	0	1
0	1	0	1	0	1	1	0
0	1	1	0	0	1	1	1
0	1	1	1	1	0	0	0
1	0	0	0	1	0	0	1
1	0	0	1	0	0	0	0
1	0	1	0	1	0	1	1
1	0	1	1	0	1	0	0
1	1	0	0	1	1	0	1
1	1	0	1	0	1	0	0
1	1	1	0	1	1	1	1
1	1	1	1	1	0	0	0

State Transition Table



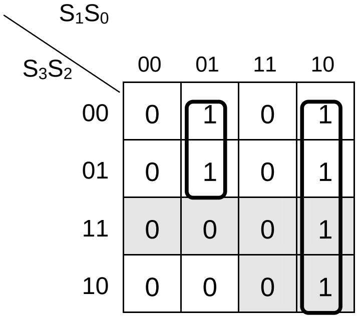
S_3S_2 \ S_1S_0	00	01	11	10
00	0	0	0	0
01	0	0	1	0
11	1	0	1	1
10	1	0	0	1

$\mathbf{NS_3 = S_3S'_0 + S_2S_1S_0}$



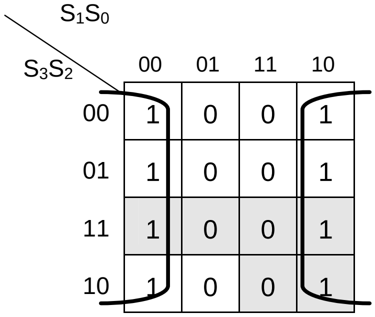
S_3S_2 \ S_1S_0	00	01	11	10
00	0	0	1	0
01	1	1	0	1
11	1	1	0	1
10	0	0	1	0

$\mathbf{NS_2 = S_2S'_0 + S_2S'_1 + S'_2S_1S_0}$



S_3S_2 \ S_1S_0	00	01	11	10
00	0	1	0	1
01	0	1	0	1
11	0	0	0	1
10	0	0	0	1

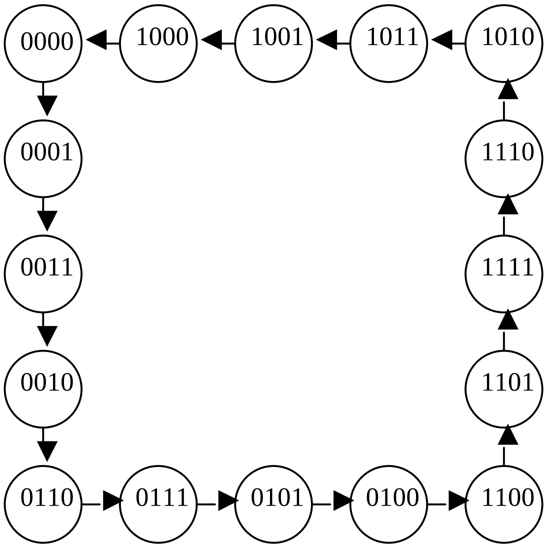
$\mathbf{NS_1 = S_1S'_0 + S'_3S'_1S_0}$



S_3S_2 \ S_1S_0	00	01	11	10
00	1	0	0	1
01	1	0	0	1
11	1	0	0	1
10	1	0	0	1

$\mathbf{NS_0 = S'_0}$

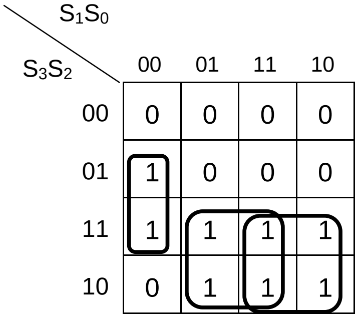
7.11 The gray-code counter cycles through all possible 4-bit codes. No self-starting transitions are needed because all possible states are already in the cycle.



State Transition Diagram

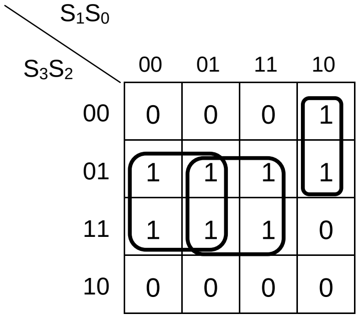
State				Next State			
S_3	S_2	S_1	S_0	NS_3	NS_2	NS_1	NS_0
0	0	0	0	0	0	0	1
0	0	0	1	0	0	1	1
0	0	1	0	0	1	1	0
0	0	1	1	0	0	1	0
0	1	0	0	1	1	0	0
0	1	0	1	0	1	0	0
0	1	1	0	0	1	1	1
0	1	1	1	0	1	0	1
1	0	0	0	0	0	0	0
1	0	0	1	1	0	0	0
1	0	1	0	1	0	1	1
1	0	1	1	1	0	0	1
1	1	0	0	1	1	0	1
1	1	0	1	1	1	1	1
1	1	1	0	1	0	1	0
1	1	1	1	1	1	1	0

State Transition Table



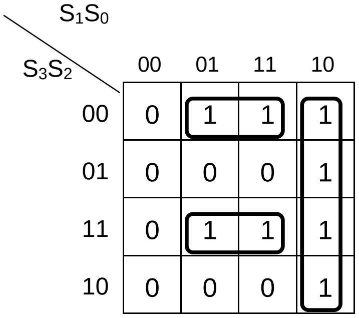
S_3S_2 \ S_1S_0	00	01	11	10
00	0	0	0	0
01	1	0	0	0
11	1	1	1	1
10	0	1	1	1

$\mathbf{NS_3 = S_3S_1 + S_3S_0 + S_2S'_1S'_0}$



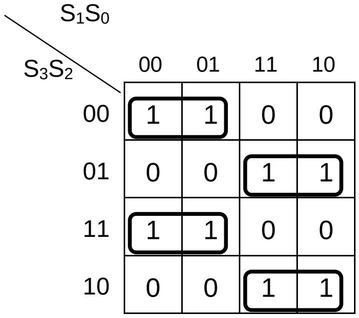
S_3S_2 \ S_1S_0	00	01	11	10
00	0	0	0	1
01	1	1	1	1
11	1	1	1	0
10	0	0	0	0

$\mathbf{NS_2 = S_2S'_1 + S_2S'_0 + S'_3S_1S'_0}$



S_3S_2 \ S_1S_0	00	01	11	10
00	0	1	1	1
01	0	0	0	1
11	0	1	1	1
10	0	0	0	1

$\mathbf{NS_1 = S_1S'_0 + S'_3S'_2S_0 + S_3S_2S_0}$



S_3S_2 \ S_1S_0	00	01	11	10
00	1	1	0	0
01	0	0	1	1
11	1	1	0	0
10	0	0	1	1

$$\mathbf{NS_0 = S'_3S'_2S'_1 + S'_3S_2S_1 \\ + S_3S_2S'_0 + S_3S'_2S_0}$$

8.5 A particular Mealy machine has 3 FFs, 2 inputs, 6 outputs.

(a) Three FFs can represent 5–8 states.

(b) With 2 inputs, there are $2^2=4$ possible input patterns. Thus there are at most 4 transitions from any given state.

(c) With 8 states, there are at most 8 transitions into any given state.

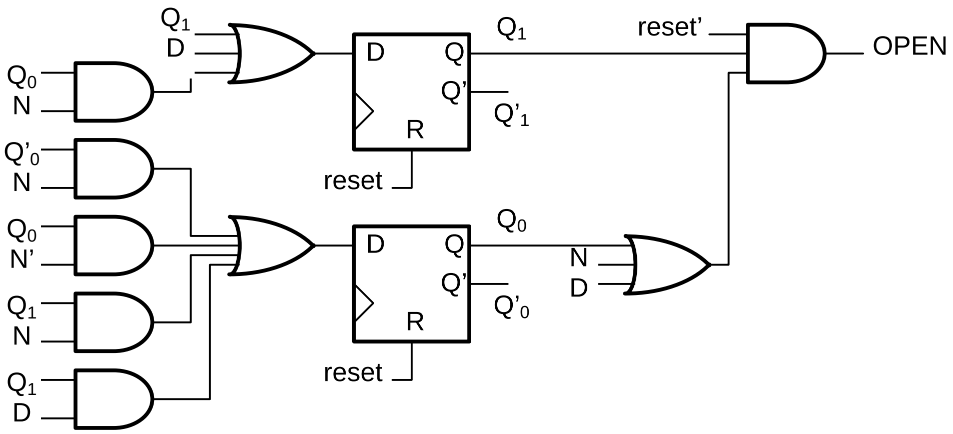
(d) Since a Mealy machine associates outputs with transitions, the maximum number of output patterns is upper-bounded by the number of transitions. There are at most 32 transitions (4 from each of 8 states), hence at most 32 output patterns. This is within the limit $2^6=64$ imposed by the number of output bits.

8.11 The Mealy state transition table is shown below. The next-state function is identical to the Moore implementation in Katz (Figures 8.13-8.15). Only the OPEN output is different. Omitted from the table is the implicit reset transition "reset/0" from each state to state 00. We hack this transition into the design by using a reset pin on the state flip-flops and by ANDing OPEN with reset' (this assumes that reset is synchronous).

Present State		Inputs		Next State		Output
Q_1	Q_0	D	N	D_1	D_0	OPEN
0	0	0	0	0	0	0
0	0	0	1	0	1	0
0	0	1	0	1	0	0
0	0	1	1	X	X	X
0	1	0	0	0	1	0
0	1	0	1	1	0	0
0	1	1	0	1	1	1
0	1	1	1	X	X	X
1	0	0	0	1	0	0
1	0	0	1	1	1	1
1	0	1	0	1	1	1
1	0	1	1	X	X	X
1	1	0	0	1	1	1
1	1	0	1	1	1	1
1	1	1	0	1	1	1
1	1	1	1	X	X	X

Q_1Q_0 \ D N	00	01	11	10
00	0	0	X	0
01	0	0	X	0
11	1	1	X	1
10	0	1	X	1

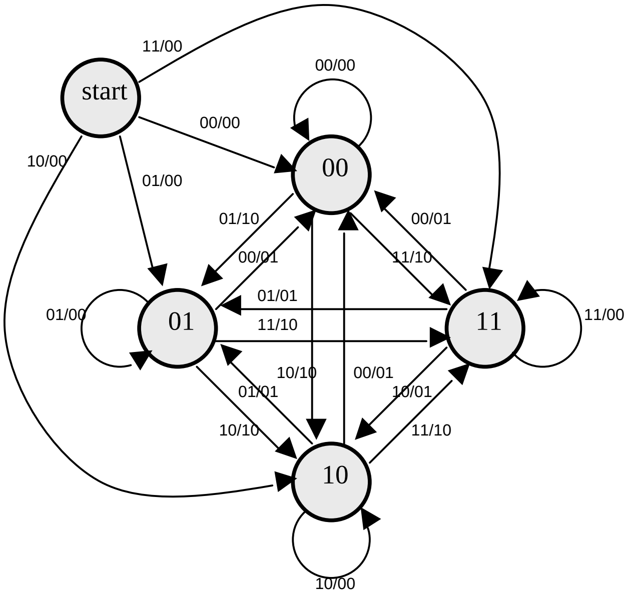
OPEN $= Q_1Q_0 + Q_1N + Q_1D$
$= Q_1(Q_0+N+D)$



The diagram above shows an asynchronous Mealy machine. A synchronous Mealy machine would look the same except that OPEN must pass through an additional flip-flop. The synchronous Mealy machine delays its outputs by a cycle, much like the Moore machine. However, while the output of the Moore machine depends only on the present state, the output of the synchronous Mealy machine still depends on the particular transition that entered the present state (*i.e.* depends on the previous state and input).

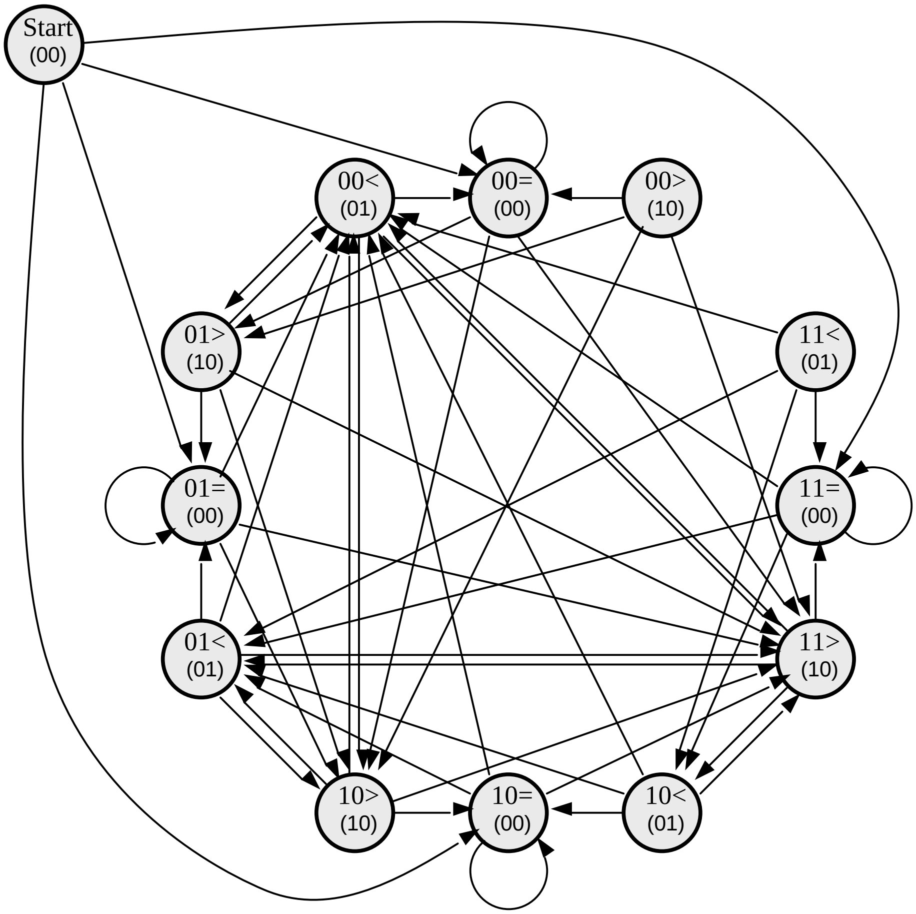
8.17 In a circuit that compares a present input with a previous input, the states must encode the value of the previous input. A Mealy design does just that – it uses 4 states to represent the previous input, plus a start state (5 states total). Since Mealy outputs are allowed to depend on inputs from the same cycle, the Mealy design can compute and output comparison results immediately. In a Moore design, the output (comparison result) must be encoded in the state and is thus delayed until the next state. Each state must encode not only the previous input but also the result of the previous comparison. For each of the 4 possible previous inputs, there are 3 possible comparison results, requiring 12 states, plus a start state (13 states total).

(a) Mealy machine.
Transition notation: X_1X_2/Z_1Z_2



(b) Moore machine.

The name of each state indicates the input used to reach it and the comparison between that input and the previous one. The output of each state is denoted as (Z_1Z_2). The inputs associated with each transition arc are omitted for brevity.


Start
(00)
00<
(01)
00=
(00)
00>
(10)
01>
(10)
11<
(01)
01=
(00)
11=
(00)
01<
(01)
11>
(10)
10>
(10)
10=
(00)
10<
(01)